Laboratorio de Modelización Virtual de la Ciudad
LMVC

Modelo interactivo Museo Marítimo de Barcelona, ITACA.

(ITACA: Plan AVANZA I+D en cooperación: Inteligencia ambienTal para ACcesibilidad al pAtrimonio)

Juan Manuel Corso Sarmiento
Arquitecto

Introducción

El modelo interactivo basado en el programa Unity 3D (herramienta de desarrollo de juegos multiplataforma), parte de la dificultad de gestión entre programas que se manejo en el proyecto PATRAC (Patrimonio accesible: I+D+i para una cultura sin barreras)[1] con programas como Quest 3d entre otros, en el cual, la iluminación, el modelo, los sensores aplicados al motor de videojuego, etc. funcionan de forma independiente y se generan en programas que pierden la flexibilidad de la construcción del modelo al ser sistemas rígidos de exportación e importación de los elementos que lo construyen, limitan la edición de la plataforma final.

En contraste Unity permite modificar el modelo, modificar mapas de UV, generar mapas de Iluminación, gestionar mejor la información del modelo base, todo ello sin salir de una única plataforma, sin necesidad de reimportar, facilitando la gestión de una base de información compleja, que evoluciona, dada la complejidad del edificio base estudiado (Museo Marítimo de Barcelona). Por otra parte esta plataforma es de uso gratuito y permite trabajar con otros sistemas operativos diferentes a Windows, permitiendo ampliar el horizonte de dispositivos móviles y táctiles sobre la cual puede funcionar el demostrador, como por ejemplo ordenadores MAC, consolas de videojuegos; Wii, hasta dispositivos móviles como iPhone, iPAD o que utilice sistema Android, lo que lo convierte en una de las plataformas más útiles.

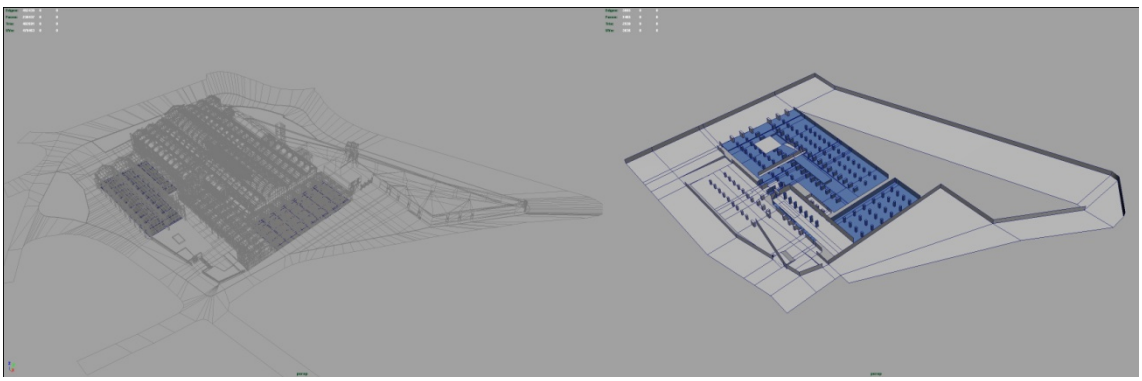
OPTIMIZACIÓN DE LA PLATAFORMA:

La optimización de la plataforma se ve reflejada en:

1. Calculo en tiempo real de propiedades físicas.

En este punto destaca las ventajas de utilizar un modelo volumétrico invisible para generar el cálculo de colisión de los Rigidbories Figura 1.

Figura 1. Modelo volumétrico que actúa como elemento de colisión, reduciendo el cálculo de procesamiento en tiempo real. Izquierda modelo de 937132 caras, derecha modelo volumétrico de 2900 caras.



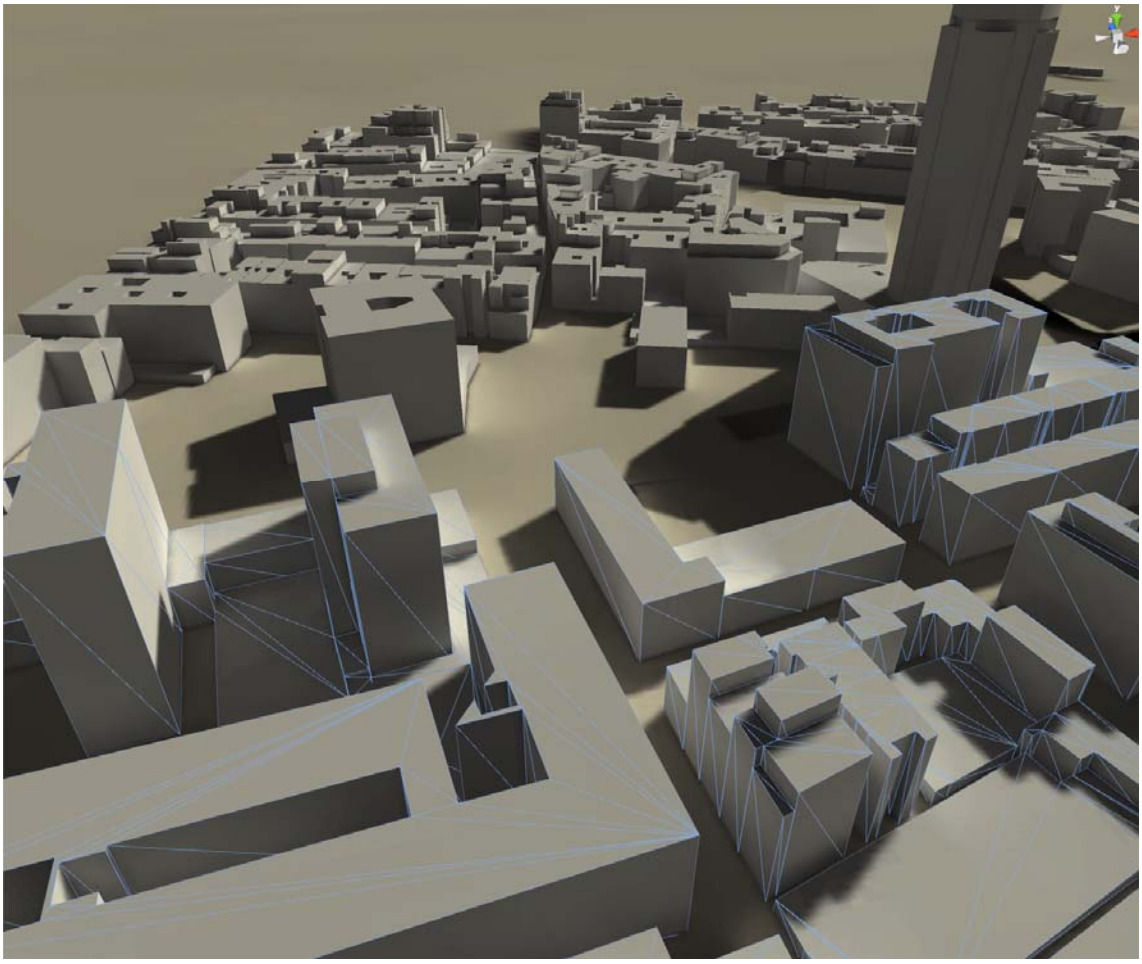
[1] Proyecto singular y estratégico del Ministerio de Educación y Ciencia, bajo el número de expediente PSE-380000-2007 finalizando su segunda anualidad con el número de expediente PSE-39-80000-2008.

Fuente: Propia

2. Optimización del modelo de contexto volumétrico.

Generado a partir la información base del catastro, (la información del ICC Instituto Cartográfico de Catalunya, se a extruido en función del número de plantas por una altura de 3 metros), eliminando planos que no son visibles, corrigiendo la unión entre vértices y reagrupando el modelo en función a los cálculos por oclusión.

Figura 2. Optimización del modelo generado con información catastral (ICC).



Fuente: Propia

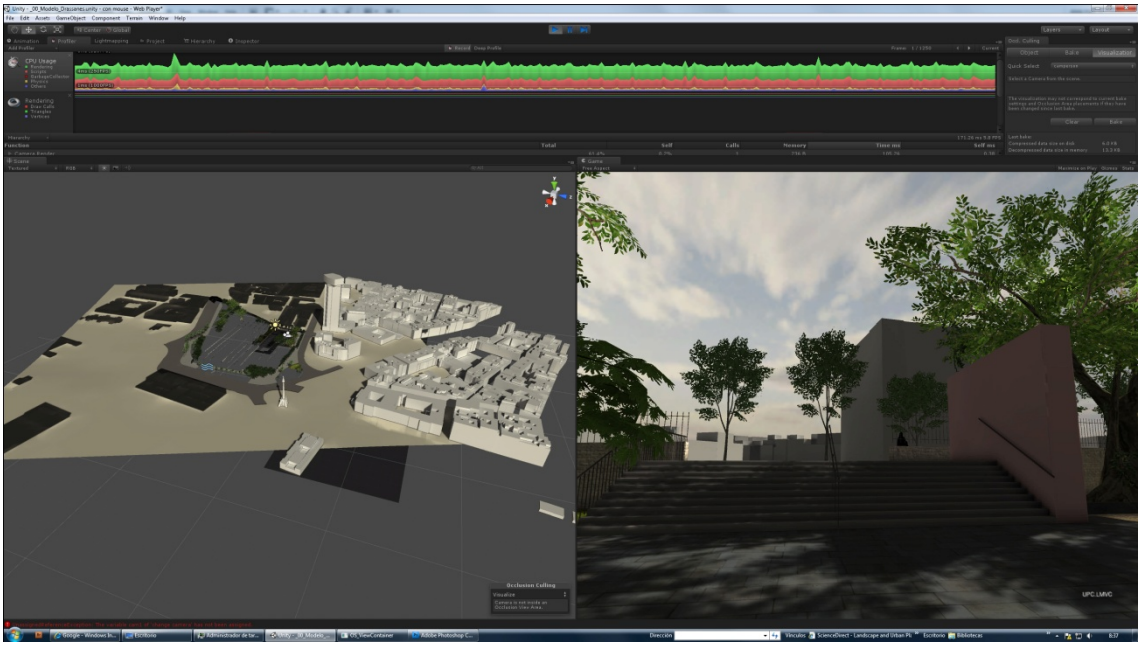
3. Reorganización de todo el modelo y gestión por oclusión.

En función ya no del programa arquitectónico como en el proyecto PATRAC, sino en función de la visualización de la cámara y las texturas, en cuanto a que solo se calcule la información que se visualiza y que está este organizada en su desdoblamiento en mapas de UV como fachadas y planos con cierta similitud a planos arquitectónicos, para complementar la información del modelo.

Un ejemplo de ello es la imagen de la figura 3, en la cual solo se visualiza la salida del jardín del Rei, por lo cual el resto del modelo desaparece en el cálculo en tiempo real, este proceso se entiende cómo gestionar la información por su oclusión.

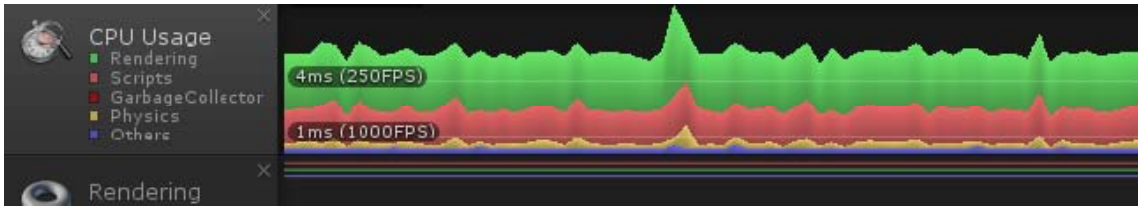
En la figura 4 vemos como se llega a un promedio de 120 cuadros por segundo, comparado con los 70 cuadros por segundo que se lograban sin esta optimización.

Figura 3. Gestion por oclusión



Fuente: Propia

Figura 4. Recursos del ordenador en tiempo real

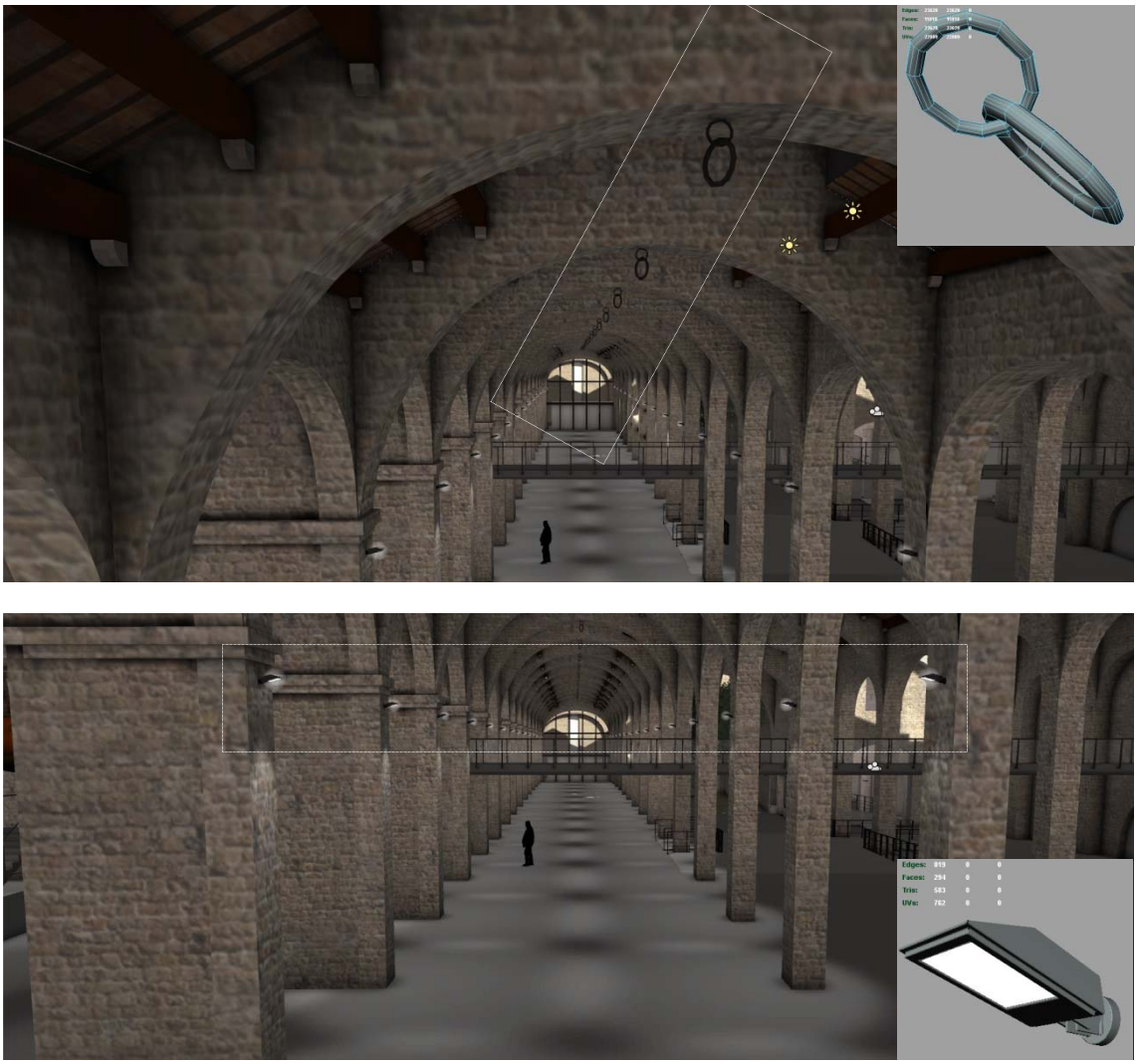


Fuente: Propia

4. Elementos prefabricados

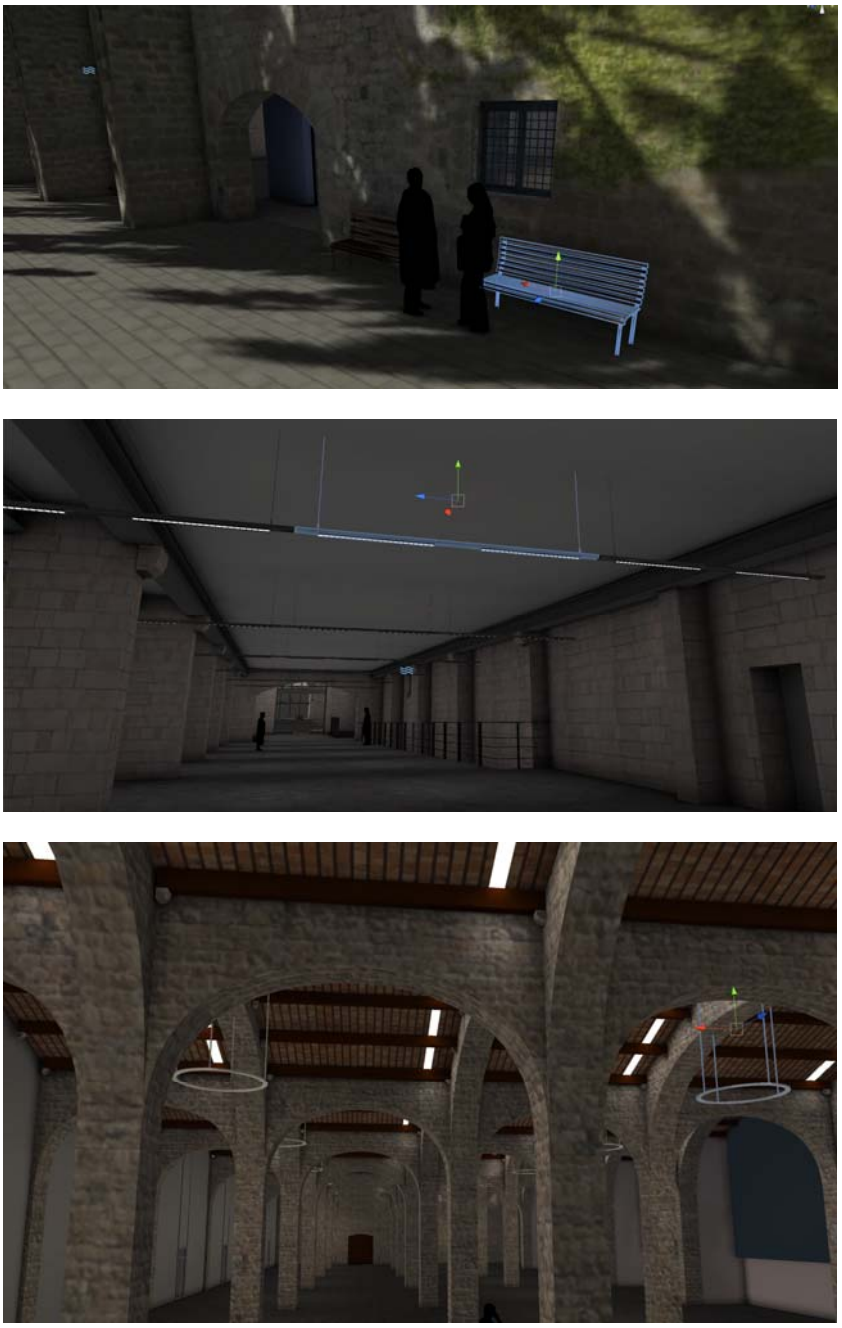
En busca de la optimización del modelo, los elementos repetitivos se eliminaron del modelo base, y se generaron modelos prefabricados, manteniendo un único modelo de referencia, el cual se repite y se desplaza a la ubicación que le corresponde, que al variar modifica todos los elementos que se basan en él.

Figura 5. Modelos prefabricados



Fuente: Propia

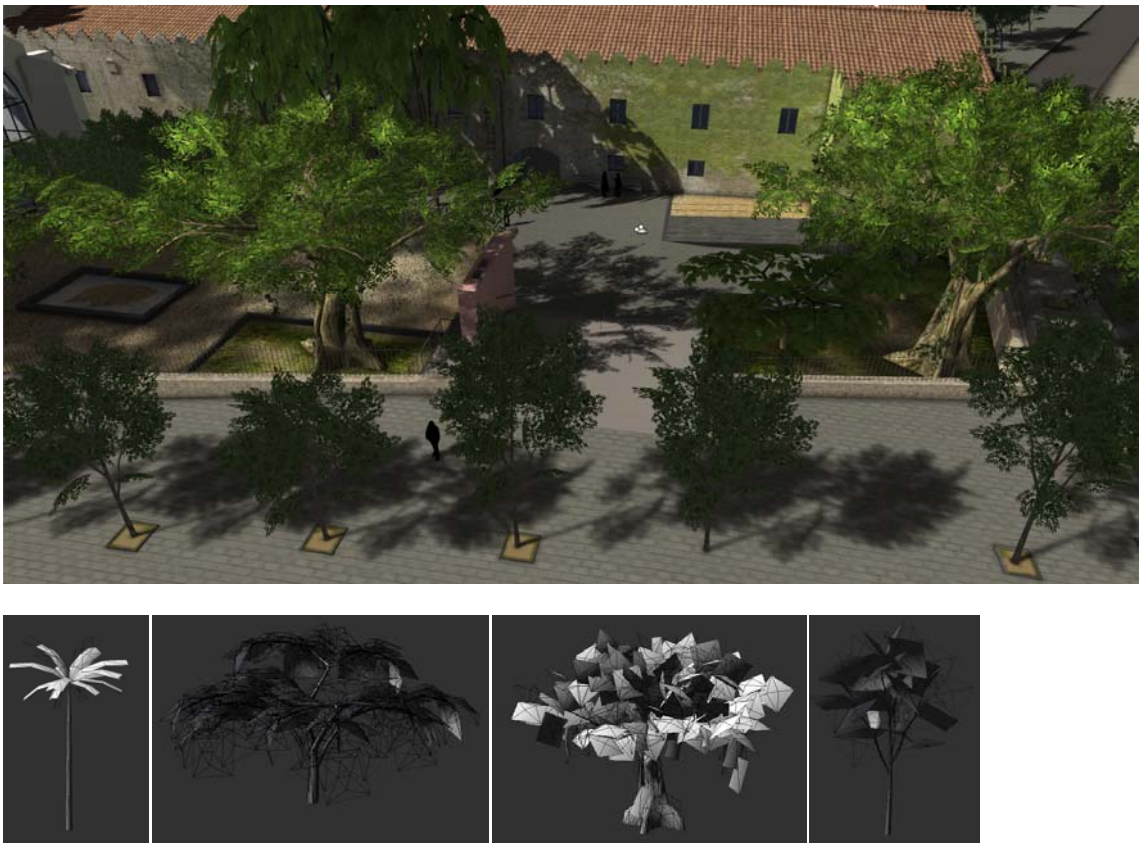
Figura 6. Modelos prefabricados



Fuente: Propia

Los elementos prefabricados también pueden tener scripts que permitan la interacción con ellos, como por ejemplo el viento que en la figura 7 mueve los arboles, también pueden tener comportamientos según la ubicación del usuario del demostrador, ya que al estar lejos, los modelos complejos como los arboles pasan de ser modelos volumétricos a imágenes fijas.

Figura 7. Modelos prefabricados aplicados sobre el terreno



Fuente: Propia

5. Optimización del Script

Una limitante que se genera en el modelo son scripts complejos como el que permite la visión en estéreo, este script se aisló en una escena independiente, que solo correrá en plataformas que no sean móviles, como las PC.

Menú lateral

```
function OnGUI () {
	GUI.Box (Rect (10,10,120,200), "OPCIONES");
	if (GUI.Button (Rect (20,40,100,30), "DRASSANES")) {
		Application.LoadLevel (1);
	}
	// Make the second button. If it is pressed, Application.Loadlevel (1) will be executed
	if (GUI.Button (Rect (20,80,100,30), "GALERA")) {
	Application.LoadLevel (2);
	}
	// Make the third button. If it is pressed, Application.Loadlevel (2) will be executed
	if (GUI.Button (Rect (20,120,100,30), "TEORICO")) {
	Application.LoadLevel (3);
	}
	// Make the last button. If it is pressed, Application.Quit will be executed
	if (GUI.Button (Rect (20,160,100,30), "EXIT")) {
	Application.Quit ();
	}
}
```

Menú con letras

```
function OnGUI () {

	if (Input.GetKeyDown (KeyCode.Alpha1)) {
	Application.LoadLevel (1);
	}
	if (Input.GetKeyDown (KeyCode.Alpha2)) {
	Application.LoadLevel (2);
	}
	if (Input.GetKeyDown (KeyCode.Alpha3)) {
	Application.LoadLevel (3);
	}
	if (Input.GetKeyDown (KeyCode.Alpha0)) {
	Application.Quit ();
	}
}
```

Comando Salir

```
function Update () {
	if (Input.GetKeyDown(KeyCode.Escape)) {
		Application.LoadLevel ("_00_Modelo_Drassanes");
	}
}
```

Iniciar acción por proximidad a un objeto

```
var enemy : Transform;
function Update() {
if ( Vector3.Distance( enemy.position, transform.position ) < 2 )
		Application.LoadLevel ("_04_Galera");
}
```

Iniciar animación

```
if (GUI.Button (Rect (12,40,15,15)," "))  {
	animation.Play ("recorrido guiado");
}
```

Menú GUI Botton

```
var mySkin : GUISkin;

function OnGUI () {
	GUI.skin = mySkin;
	if (GUI.Button (Rect (12,40,15,15)," "))  {
		Application.LoadLevelAdditive ("inicio");
	}
}
```

Cambiar de color elementos con el paso del mouse mostrando nombre del elemento

```
var rolloverText = "";
var rolloverColor = Color.red;
private var originalColor : Color;
private var textDisplay : GUIText;

function Start() {
   originalColor = renderer.material.color;
   if (rolloverText == "") { rolloverText = gameObject.name; }
   textDisplay = GameObject.Find("Rollover Text Display").guiText;
}
```

```
function OnMouseEnter() {
    Debug.Log(textDisplay.gameObject.name);
    textDisplay.text = rolloverText;}

function OnMouseExit() {
    Debug.Log(textDisplay.gameObject.name);
    textDisplay.text = "";
}
```

6. Interacción entre Escenas

La interacción entre escenas permite dividir la información entre archivos a ejecutar en tiempo real, ya que solo se carga una escena a la vez. Por una parte está el modelo interactivo del museo (figura 8), al acercarse al punto de la galera cambia al escenario de la galera en el mar, mediante el script de acción por proximidad, lo cual se ve en la figura 9 y 10.

VISUALIZACIÓN DE LA APLICACIÓN

Figura 8. Detalle alcanzado en el modelo interactivo.

Fuente Propia

Figura 9. Por proximidad al plano con la imagen del Barco cambia de escena a un entorno donde se aprecia la Galera en el mar.

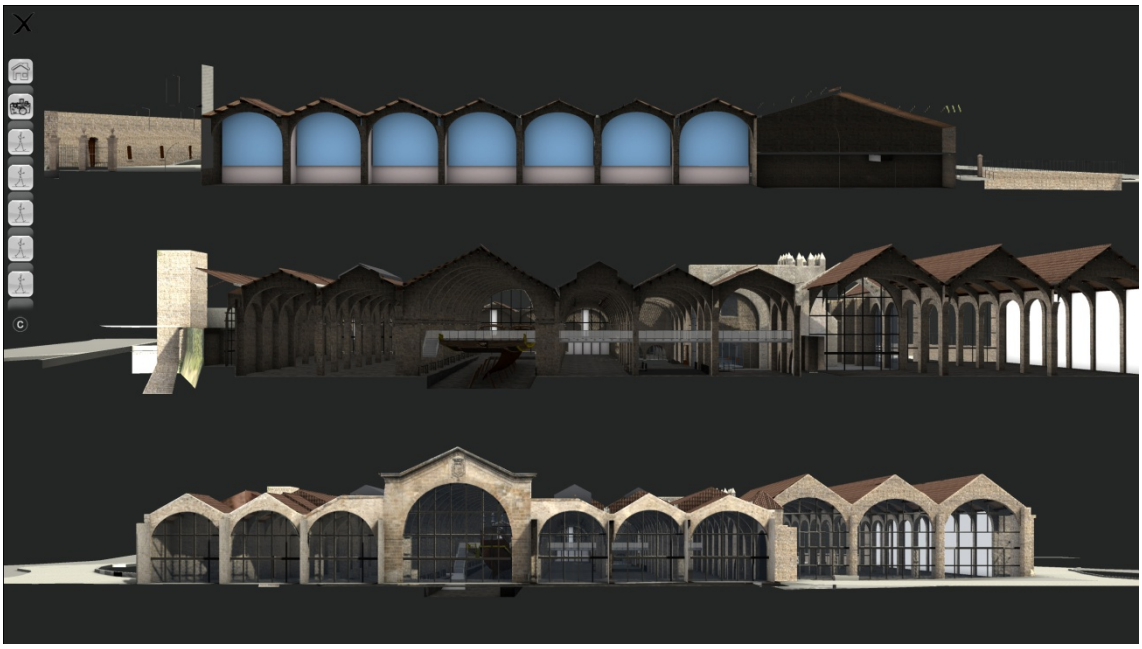
Fuente Propia

Figura 10. Interacción con la Galera

Fuente Propia

De igual forma se cambia de escena al cambiar el tipo de cámara, ya que la cámara de estéreo consume memoria en su script que retrasa el procesamiento en tiempo real, limitando este efecto a un modelo menor. Este problema no sucede en todas las cámaras utilizadas, como por ejemplo en la figura 11, con la cámara que secciona a el modelo haciendo cortes de la visualización según un rango preestablecido, que puede controlar el usuario, manejando los elementos de visualización ya no por escena sino por capas de renderizado, que controlan lo que es visible y lo que no para cada cámara.

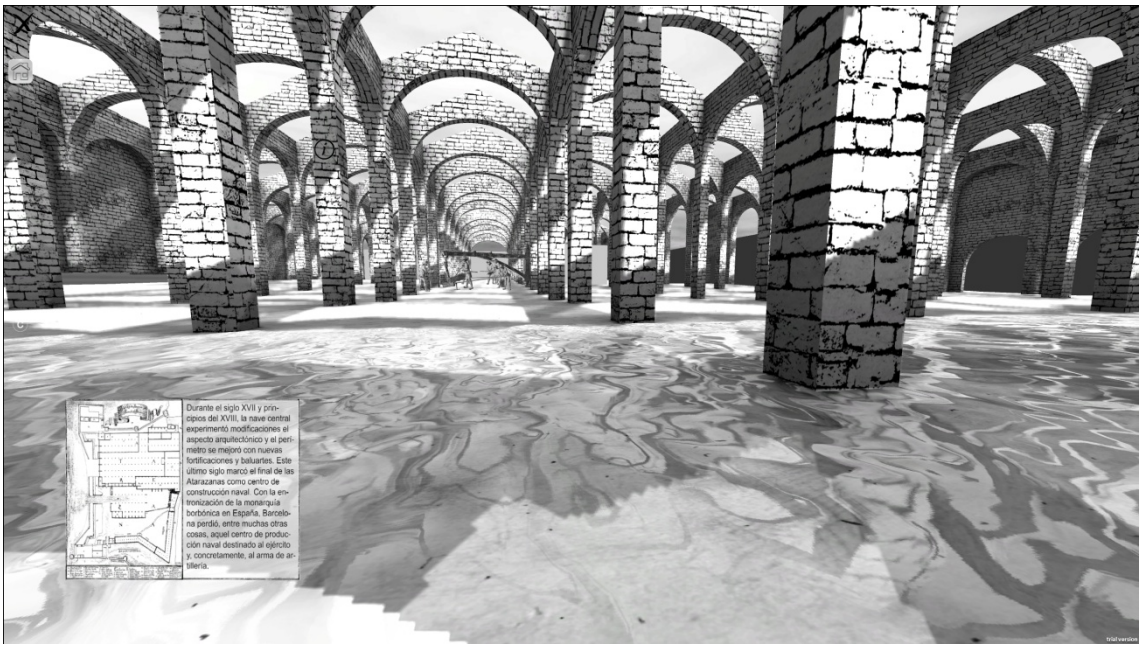
Figura 11. Cámara que secciona el modelo teórico de forma interactiva.



Fuente Propia

Paralelo a este ejemplo de interacción entre escenas se generan escenarios teóricos que explican etapas del modelo, entendiendo que es una representación teórica, se cambia la escena y los modelos cambian su tipo de expresión, tanto en la iluminación como en el texturizado, figura 12.

Figura 12. Modelo de las Naves medievales antes de que se modificaran las dos naves centrales por una nave de mayor tamaño y como el agua del mar entraba en estas.



Fuente. Propia